САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА МОДЕЛИРОВАНИЯ ЭКОНОМИЧЕСКИХ СИСТЕМ

Патин Михаил Владиславович

Выпускная квалификационная работа бакалавра

Сравнительный анализ дескрипторов особых точек изображений с внедрением алгоритмов под операционной системой «Android»

Направление 010400.62

Прикладная математика и информатика

Научный руководитель,
кандидат физ.-мат. наук,
доцент
Ковшов А.М.

Санкт-Петербург

2016

Оглавление

Введение

Уже несколько десятилетий ученые из разных стран занимаются разработкой алгоритмов, позволяющих научить компьютер видеть так же, как видит сам человек. Если для людей получать необходимую информацию посредством зрительного канала является чем-то простым и само собой разумеющимся, то обучить компьютер подобным вещам является и по сей не выполнимой задачей. Множество IT корпораций работают над её решением, но это требует большого вложения человеческого труда, финансовых затрат и вычислительных мощностей.

Но существуют методы, которые позволяют, хоть и при использовании в узконаправленных задачах, получить желаемый результат при меньших затратах. Они основаны не на структуре человеческого аппарата анализа и интерпретации изображений, а непосредственно на особенностях самого изображения. Одни из таких методов основаны на нахождении особых точек и их численного описания, на которые люди даже не обращают внимания. Основываясь только на наборе таких данных цифрового изображения можно с достаточно высокой точностью позволить компьютеру работать с визуальными образами подобно человеку. Вопрос эффективности таких алгоритмов ставится наиболее остро при работе на мобильных устройствах – смартфонах, без которых сложно представить образ современного человека.

С помощью мобильных устройств люди делают тысячи фотографий каждый день, нередко они получаются плохого качества и возникает необходимость делать повторные снимки. Со временем это может привести к засорению памяти из-за накопления большого количества похожих фотографий.

В данной работе сравниваются несколько современных методов поиска особых точек и расчета их дескрипторов, по результатам их применения в классификаторе, объединяющего в группы снимки по степени сходства.

Обзор литературы

Разработкой алгоритмов, обнаруживающих особые точки изображений и описывающие определенные их свойства, занимаются достаточно давно. Внедряются новые подходы для решения проблем скорости работы и качества находимых особенностей. Так же проводятся сравнения между методами по различным критериям.

В [1] рассматривается алгоритм ORB (**O**riented FAST and **R**otated **B**RIEF), представляющий из себя комбинацию из модифицированных алгоритмов нахождения особых точек с помощью FAST [4] и последующим определением их особенностей в виде бинарной строки по модифицированному методу BRIEF [5]. Данный подход дает, по результатам их тестирования, значительный выигрыш в скорости при сопоставимой или лучшей точности, чем SIFT (**S**cale **I**nvariant **F**eature **T**ransform) [8] и SURF (**S**peeded **U**p **R**obust **F**eatures) [9] соответственно.

В работе [2] представлен алгоритм BRISK (**B**inary **R**obust **I**nvariant **S**calable **K**eypoints), в котором особые точки обнаруживаются с использованием FAST [4] при линейном изменении масштаба начального изображения. Дескрипторы описываются в бинарном виде по улучшенному алгоритму BRIEF [5]. В сравнительных тестах получена сравнимая с SIFT [8] и SURF [9] точность на различном типе изображений, но при этом скорость работы алгоритма, по их результатам, в разы выше.

В [7] представлен алгоритм A-KAZE, в котором поиск особых точек осуществляется при нелинейном масштабировании изображения с использованием схемы FED (**F**ast **E**xplicit **D**iffusion) [8]. В качестве бинарного дескриптора применяется M-LDB (**M**odified-**L**ocal **D**ifference **B**inary). Результаты работы алгоритма, по данным приведенных исследований, превосходят по всем параметрам ORB, BRISK, SIFT и SURF.

Цель

Целью данной работы является проведение сравнительного анализа методов ORB, BRISK, AKAZE, нахождения особых точек изображения и их дескрипторов. Для достижения поставленной цели предполагается разработать прототип приложения под мобильные устройства под ОС “Android”.

Решение данной задачи требует реализации следующих этапов:

1. Рассмотреть методы поиска особых точек и дескрипторов.
2. Применить алгоритмы сопоставления дескрипторов изображений.
3. Разработать критерий сходства изображений по содержанию.
4. Разработать алгоритм распределения по группам коллекции фотографий.
5. Создать программную реализацию для проведения исследований.
6. Провести оценку результатов на различного типа фотографиях и параметрах алгоритма сопоставления дескрипторов.

Глава 1: Алгоритмы обнаружения особых точек и их дескрипторов

Для обнаружения признакового описания изображения необходимо привязываться к его локальным особенностям - особым точкам.

Процесс поиска особых точек осуществляется с помощью *детектора*.

Особая точка, или *особенность* – это точка изображения, удовлетворяющая ряду свойств:

1. *Определенность (distinctness)* – особенность должна выделяться на фоне среди соседних точек.

2. *Устойчивость (repeatability)* – изменение яркости, контрастности и цветовой гаммы не должны влиять на место особой точки на объекте или сцене.

3. *Инвариантность (invariance)* – особые точки должны обладать устойчивостью к повороту, изменению масштаба изображения и смене ракурса съемки.

4. *Стабильность (stability)* –зашумленность изображения, не превышающая определенный порог, не должна влиять на работу детектора.

5. *Интерпретируемость (interpretability)* – особые точки должны быть представлены в формате, пригодном для дальнейшей работы.

6. *Количество (quantity)* – количество обнаруженных особых точек должно обеспечивать требуемому их количеству для обнаружения объектов.

Дескриптор (от лат. *descriptor* — описывающий) – описание особой точки, определяющее особенности её окрестности, представляет собой числовой или бинарный вектор определенных параметров. Длина вектора и вид параметров определяются применяемым алгоритмом. Дескриптор позволяет выделить особую точку из всего их множества на изображении, это необходимо для составления ключевых пар особенностей, принадлежащих одному объекту, при сравнении разных изображений.

В связи с патентными ограничениями алгоритмов SIFT [12] и SURF [13], их не удалось реализовать в приложении для проведения сравнительных тестов. Поэтому далее рассматриваются только одни из свободно распространяемых ORB, BRISK, A-KAZE. Их выбор основан на том, что они имеют бинарный вид дескриптора. Такая структура обеспечивает высокую скорость при их сравнениях, что сделало их достаточно популярными.

1.1 ORB – Oriented FAST and Rotated BRIEF

ORB представлен в 2011г [1]. В его основе лежит комбинация таких алгоритмов как детектор FAST (**F**eatures from **A**ccelerated **S**egment **T**est) [4] и дескриптор BRIEF (**B**inary **R**obust **I**ndependent **E**lementary **F**eatures) [5] с некоторыми улучшениями.

Детектор FAST.

Для поиска угловых точек поочерёдно рассматриваются окрестности по 16 пикселей вокруг каждого пикселя p.

Точка p считается подозрительной на особую, если существует N пикселей (в данной работе $N=9$) в её окружности длиной 16 пикселей, если все N ярче $I_P + t$ или темнее $I_P - t$, где I_P – яркость точки p, t –пороговая величина. При выполнении этого условия далее исследуется значения яркости на окружности под номерами 1, 5, 9, 13 (рис. 1.1). Если для трех пикселей из четырех выполняется условие $I_i < I_P - t$ или $I_i > I_P + t$, $i = 1 \dots 4$, тогда p считается особой точкой.

Выбор только 4 пикселей на окружности позволяет быстро отсеять не подходящие точки, но в некоторых случаях возможно определение разных особенностей в одной окружности. В алгоритме ORB максимальное количество особых точек по умолчанию не более 500, если их больше, то к ним применяется детектор углов Харриса [10], для исключения наименее значимых.

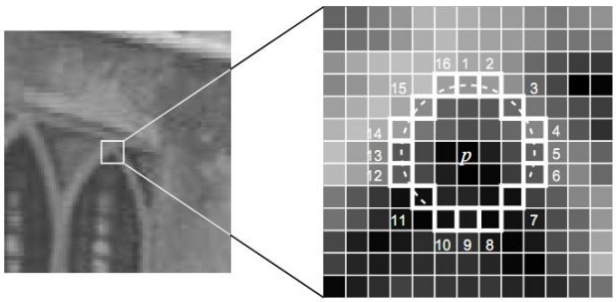


Рис. 1.1. Рассматриваемая окрестность точки *p* FAST детектора [1].

Для инвариантности к масштабированию применяется вышеописанный алгоритм на пирамиде Гаусса. Октавами c_i которой является изначальное изображение c_0 сжатое с линейным шагом.

Введение параметра угловой ориентации позволяет добиться устойчивости детектирования при вращении объекта. Он основан на направлениях градиента яркости относительно центра точки, направление с наибольшей интенсивностью назначается ориентацией особой точки θ.

Дескриптор направленный BRIEF.

Данный дескриптор представляется в виде вектора длиной 256, состоящего из результатов бинарных тестов вокруг особой точки. В окрестности 31×31 пиксель сравниваются средние значения яркостей между x и y, где x, y – области 5×5 пикселей:

$$\tau(I; x, y) \coloneqq \begin{cases} 1 : I_x < I_y \\ 0 : I_x \geq I_y \end{cases}; \; I \; - \textit{средняя яркость выбранной области.}$$

Для достижения инвариантности к вращению область вычисления дескриптора ориентируется по ориентации особой точки θ.

Все $n = 256$ наборов x_i и y_i формируют матрицу S размерностью $2 \times n$. Далее S с помощью матрицы поворота R_θ ориентируется в соответствии с углом θ:

$$S_\theta = R_\theta S.$$

А сам вектор дескриптора записывается как:

$$g_n(I, \theta) \coloneqq f_n(I) | (x_i, y_i) \in S_\theta,$$
$$\textit{где} \;\; f_n(I) \coloneqq \sum_{1 \leq i \leq n} 2^{i-1} \tau(I; x_i, y_i).$$

1.2 BRISK – Binary Robust Invariant Scalable Keypoints

Данный метод представлен в 2011г. Детектирование особых точек осуществляется с помощью FAST (**F**eatures from **A**ccelerated **S**egment **T**est), а дескриптор BRIEF, но в их работу были внесены некоторые изменения.

Поиск особых точек.

Для достижения инвариантности к масштабу, предлагается выбирать наилучшую особую точку с максимальным значением интенсивности в пирамиде, которая состоит из 4 октав c_i и 4 внутренних октав d_i, $i = 0\ldots3$. Октавы формируются как сжатие оригинального изображения c_0 в 2^i раза. Внутренние октавы расположены между c_i и c_{i+1} и представлены в виде сжатой c_0 в $\frac{2}{3}2^i$ раза. Поиск особых точек в октавах осуществляется детектором FAST.

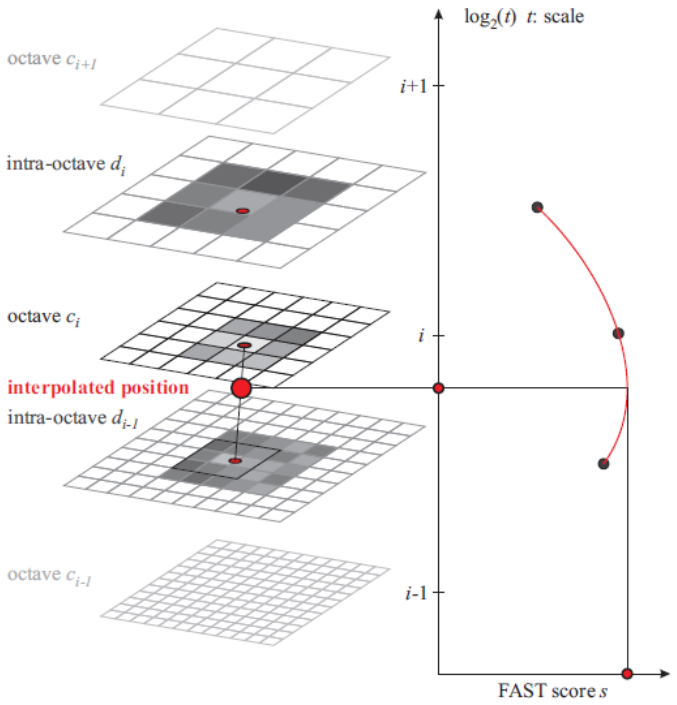


Рис. 1.2. Пример поиска особой точки с максимальным значением S [2].

Дескриптор BRISK.

Область вокруг особой точки разбивается на 60 участков p (рис. 1.3):

$$\mathcal{A} = \{(p_i, p_j) \in R^2 \times R^2 | i < N \wedge j < i \wedge i, j \in \mathbb{N}\}$$

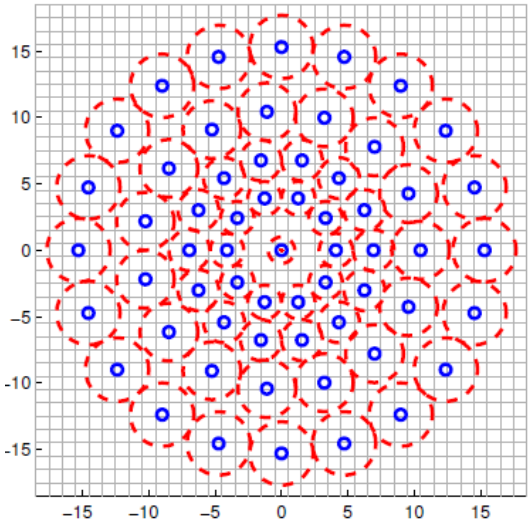


Рис 1.3. Область вычисления дескриптора [2].

Множество $\mathcal{A}$ разбивается на 2а подмножества:

$$\mathcal{S} = \{(p_i, p_j) \in \mathcal{A} | \|p_j - p_i\| < \delta_{max}\} \subseteq \mathcal{A}$$
$$\mathcal{L} = \{(p_i, p_j) \in \mathcal{A} | \|p_j - p_i\| > \delta_{min}\} \subseteq \mathcal{A}$$

где $\delta_{min} = 13.67t,\ \delta_{max} = 9.75t,\ t -$ *размер особой точки.*

Вычисляется среднее значение градиента множества $\mathcal{L}$:

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{|\mathcal{L}|} * \sum_{(p_i, p_j) \in \mathcal{L}} \left[(p_j - p_i) \frac{I(p_j, \sigma_j) - I(p_i, \sigma_i)}{\|p_j - p_i\|^2} \right]$$

Дескриптор состоит из бинарной строки длиной 512, заполненной результатами проведенных тестов в множестве $\mathcal{S}$:

$$b = \begin{cases} 1, I(p_j^\alpha, \sigma_j) > I(p_i^\alpha, \sigma_i) \\ 0, \text{иначе} \end{cases};\ \forall (p_i^\alpha, p_j^\alpha) \in \mathcal{S},$$

где $I(p_i^\alpha, \sigma_i)$ *интенсивность окрестности радиуса* σ_i *точки* p_i,

$\alpha = arctan2(g_y, g_x) -$ *угол направления градиента* g.

1.3 AKAZE – Accelerated-KAZE

При разработке данного метода, представленного в 2013 году, старались добиться высокой скорости работы как детектора, так и дескриптора. При этом найденные особые точки и их дескрипторы должны были удовлетворять высоким показателям точности при сравнении изображений.

Применение алгоритма FED - Fast Explicit Diffusion [6] на пирамидальной схеме позволяет построить нелинейную многомасштабную пирамиду. Применение нелинейного коэффициента масштабирования позволяет увеличить скорость нахождения нужной особой точки по сравнению с Гауссовой пирамидой:

Вычисление данного коэффициента основано на изменении яркости изображения при масштабировании.

Детектор.

Для каждого октавы L^i в пирамиде вычисляется определитель Гессиана.

$$L^i_{Hessian} = \sigma^2{}_{i,norm}\left(L^i_{xx}L^i_{yy} - L^i_{xy}L^i_{xy}\right),$$

где $\sigma_{i,norm} = \frac{\sigma_i}{2^{o^i}}$ *нормализированный относительно масштаба коэффициент, для вычисления* $L^i_{Hessian}$ *с учетом размера октавы* σ_i.

Производные второго порядка вычисляются с помощью фильтра Шарра[11] с шагом $\sigma_{i,norm}$. Данный фильтр позволяет учитывать ориентацию особых точек. С помощью такого подхода ищем такие точки в октаве, значение фильтра которых выше заданного порога и является наибольшим из окрестности точки 3×3 пикселей.

Далее, для каждой точки из потенциальных максимумов сравнивается её значение относительно результатов в соседних октавах $i + 1$ и $i - 1$ в окне размером $\sigma_i \times \sigma_i$ соответственно. В итоге расположение особой точки оценивается с субпиксельной точностью соответствуя квадратичной функции к определителю Гессиана в 3×3 соседних пикселей для поиска максимума.

Дескриптор M-LDB.

Первоначальный дескриптор LDB [9] основывался на тех же принципах что и рассмотренный выше BRIEF, но к сравнениям яркостных показателей областей добавили сравнение значений градиентов яркости по оси x и y, в итоге результат одного теста состоит из трех битов вместо одного. Проведение тестов проводилось в окне размером 20×20 пикселей, деленном на 4, 9 и 16 областей.

Но LDB имеет недостатки такие как не инвариантность к вращению и масштабированию. И в качестве решения этих проблем в AKAZE используется его улучшенная версия – M-LDB:

1. Окно дескриптора ориентируется по ориентации особой точки.

2. Инвариантность к масштабу получена с помощью выбора размера окна дескриптора в зависимости от размера октавы σ_i в которой найдена его особая точка.

В отличии от LDB в M-LDB тесты проводятся не между средним значением всех пикселей в области, а между заданным их количеством в зависимости от размера σ_i. Что позволяет ускорить вычисление дескриптора.

Итоговый бинарный дескриптор имеет длину 486 по три составляющих.

Глава 2: Сравнение изображений

С помощью вышеописанных методов, был получен набор дескрипторов по каждому изображению из коллекции, далее необходимо их сопоставить для поиска схожих.

2.1 Расстояние между дескрипторами

Для сравнения пары изображений в основном используют метод сравнения основанный на вычислении расстояний всех возможный пар дескрипторов $\rho(d_i, d_j{}')$.

$$\begin{cases} d - \text{дескриптор первого изображения, вектор из признаков } \alpha_k \\ d' - \text{дескриптор второго изображения, вектор из признаков } \alpha'_k \end{cases}$$

Где $\forall\, d_i \in \mathcal{D}, \forall\, d'_j \in \mathcal{D}',\ i = 1 \dots |\mathcal{D}|,\ j = 1 \dots |\mathcal{D}'|$, размерность вектора признаков $|\mathcal{K}|$ определяется в зависимости от используемого метода описания точки.

Для определения расстояний обычно используется евклидова метрика:

$$\rho(d_i, d'_j) = \sum_{k=0}^{|\mathcal{K}|} |\alpha_k - \alpha'_k|^2$$

Однако она применима только для дескрипторов, описываемых количественными переменными. Но рассматриваемые в данной работе методы, представляют описание особой точки в виде бинарной строки. И в таком случае рекомендуется применять расстояние Хемминга, которое вычисляется как количество не равных значений в векторах:

$$\rho(d_i, d'_j) = \sum_{k=0}^{|\mathcal{K}|} I(\alpha_k \neq \alpha'_k)$$

Ближайший сосед

Далее для каждого дескриптора d_i выбираются два ему ближайших d'_j и наоборот. Если у выбранного d уже есть соответствующие ему два

дескриптора, то он пропускается и поиск продолжается. В итоге каждому дескриптору d_i будут соответствовать не больше двух взаимно ближайших из $\mathcal{D}'$.

Вводится параметр отношения длин $\upsilon = \frac{\rho_{i_1}}{\rho_{i_2}}$ $(\rho_{i_1} < \rho_{i_2})$, по которому отсеваются дескрипторы не удовлетворяющие необходимому уровню определенности. Если υ больше заданного порога υ_{max}, то d_i далее не рассматривается, иначе для d_i ставится в соответствие дескриптор d_j с расстоянием ρ_{i_1}.

2.2 Алгоритм RANSAC – Random sample consensus

Отфильтровать дескрипторы только по дистанции недостаточно для достижения высокой точности определения схожих объектов на изображениях. Если объект переместился на сцене или снят с другого ракурса, то при применении трансформации «наложения» n точек одного изображения на соответствующие по ближайшему соседу n точек другого, можно выявить особенности, не относящиеся к общему объекту и тем самым уменьшить количество ложно определенных связей.

Схема работы алгоритма RANSAC заключается в циклическом повторении поиска матрицы трансформации H между случайно выбираемыми четырьмя особыми точками s_i на одном изображении и соответствующе им четырём точкам на втором:

$$s_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix}$$

Лучшая матрица трансформации считается той в которой достигнут минимум суммы отклонений всех особых точек изображений при преобразовании H, за заданное количество циклов (≤ 2000):

$$\sum_i \left[(x_i - \frac{h_{11}x'_i + h_{12}y'_i + h_{13}}{h_{31}x'_i + h_{32}y'_i + h_{33}})^2 + (y_i - \frac{h_{21}x'_i + h_{22}y'_i + h_{23}}{h_{31}x'_i + h_{32}y'_i + h_{33}})^2 \right]$$

В итоговое множество $srcPoints'$ включаются только те точки $srcPoints_i$ отклонение которых не превосходит заданного порога:

$$\|dstPoints_i - H * srcPoints_i\| < reprojThreshold$$

Где $srcPoints$ – множество всех особых точек на первом изображении, а $dstPoints$ соответствующие им особые точки на втором.

2.3 Критерий сходства изображений.

Для того чтобы определить действительно ли на изображениях расположен один и тот же объект был разработан критерий сходства:

1) Рассматривается множество $srcPoints'$, если $|srcPoints'| < 6$, то изображения не сравниваются. Иначе выполняются следующие шаги.

2) Вычисляется шаг обхода $step = \frac{|srcPoints'|}{3}$

3) В цикле $n = 0 \dots 2$:

1. Выбираются четыре особых точки $srcPoints'_j$, $j = step * i + n$, $i = 0 \dots 3$, в качестве вершин многоугольника P_{n_1} и соответствующий ему P_{n_2} из вершин $dstPoints$. Если $j \geq |srcPoints'|$, то $j = |srcPoints'| - n - 1$.

2. Вычисляются суммы длин сторон треугольников из которых состоят P_{n_1} и P_{n_2} соответственно, по четыре на каждый из многоугольников (рис.2.1).

$$tr_{0\dots3_1} = \sum_{0 \leq c < l}^{3} \sqrt{\begin{matrix} \left(srcPoints'_{c_x} - srcPoints'_{l_x}\right)^2 + \\ + \left(srcPoints'_{c_y} - srcPoints'_{l_y}\right)^2 \end{matrix}}$$

$$tr_{0\dots3_2} = \sum_{0 \leq c < l}^{3} \sqrt{\begin{matrix} \left(dstPoints_{c_x} - dstPoints_{l_x}\right)^2 + \\ + \left(dstPoints_{c_y} - dstPoints_{l_y}\right)^2 \end{matrix}}$$

3. $avargeTrDiv_n = \sum_{i=0}^{3} \frac{tr_{i_1}}{tr_{i_2}} / 4$.

4. $averageDeviation_n = \sum_{i=0}^{3}\left(avargeTrDiv_n - \sum_{i=0}^{3}\frac{tr_{i_1}}{tr_{i_2}}\right)/4.$

4) Если $avergeDeviation = min_n(averageDeviation_n) < 0.2,$ то $alikeValue = (1 - 1{,}5 * avergeDeviation) * |srcPoints'|$, иначе $alikeValue = 0.$

Экспериментальным путем было выделено оптимальное пороговое значение $avergeDeviation < 0.2,$ если неравенство выполняется то изображения считаются похожим по содержанию.

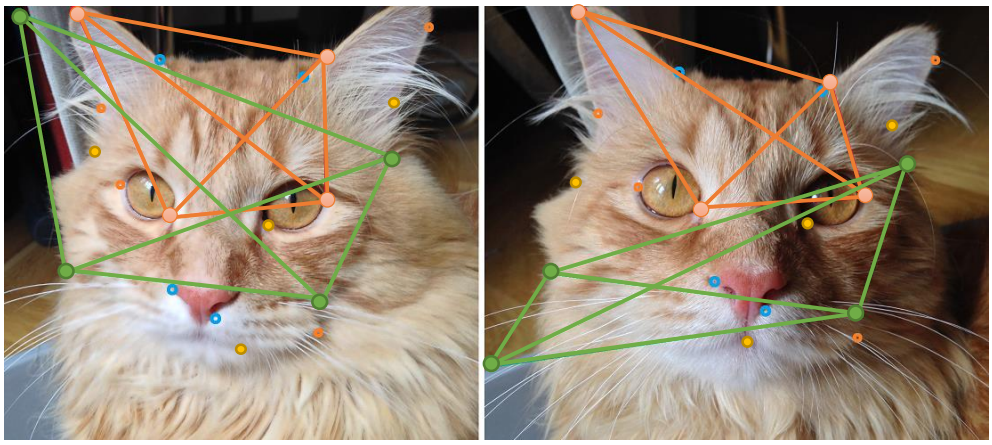


Рис 2.1. Пример работы алгоритма.

Данный подход достаточно точно позволяет определить сходство двух изображений при условии корректного сопоставления дескрипторов.

Параметр $alikeValue$ определяет степень сходства изображений, он учитывает количество связей и их геометрические отклонения $avergeDeviation$ между изображениями.

2.4 Группировка по степени сходства

Для решения задачи группировки, был разработан классификатор, основывающийся на значениях $averageDeviation$ и $alikeValue$.

1) Рассматривается симметричная матрица $k \times k$, $k -$ количество входных изображений. Необходимо вычислить $k * \frac{k-1}{2}$ попарных сравнений для её заполнения. Главная диагональ матрицы заполняется значениями $alikeValue = -1$ и $averageDeviation = 100.$ Каждому изображению

присваивается номер от 0 до $k-1$. Строки матрицы $i = 0 \dots k-1$, состоят из ячеек $j = 0 \dots k-1$, содержащих информацию о значениях $averageDeviation$, $alikeValue$ и номере изображения j, с которым проводилось сравнение изображения i.

2) Далее каждая строка сортируется по убыванию $alikeValue$.

Рассмотрим алгоритм группировки:

В цикле $i = 0 \dots k-1$, построчно:

1) Если изображение i не имеет группы, то выбирается первая ячейка i_1.
2) Если у изображения с номером i_1 в первой ячейке стоит номер i:
    1. Если $averageDeviation < 0.2$, то i и i_1 формируют одну группу.
3) Иначе если у i_1 в первой ячейке номер j, то сравниваются i и j:
    1. Если между i и j $avergeDeviation < 0.2$:
        а) Если i_1 и j не имеют группы, то i, i_1 и j формируют одну группу.
        б) Иначе i назначается в группу к i_1.
4) Иначе если $avergeDeviation > 0.2$, то i выбывает из рассмотрения и считается что группа не найдена.
5) Рассматривается следующее значение i.

Пример работы алгоритма рассмотрен в приложении 1.

Глава 3: Программная реализация

Для проведения исследований написано приложение, для мобильных устройств под ОС “Android”, с использованием языка Java.

Рассмотренные алгоритмы поиска особых точек, вычисление дескрипторов и их сопоставление на изображениях реализованы с помощью библиотеки OpenCV 3.1.0 [14]. Реализована возможность сохранения результатов работы методов ORB, BRISK, AKAZE для каждого изображения в файл в формате JSON [15].

Для разработки использовалась интегрированная среда Android Studio 2.1, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains. Необходимая минимальная версия ОС на смартфоне для установки приложения – Android 4.4 KITKAT.

Тестирование работы приложения проводилось на ноутбуке под ОС Windows 10 с процессором Intel(R) Core(TM) i5-6300HQ 2.30 ГГц с использованием эмулятора Android - Genymotion (Fast And Easy Android Emulator) версии 2.6.0 для некоммерческого использования, и смартфоне Samsung Galaxy S4 mini, на чипсете Qualcomm Snapdragon 400 с процессором Krait 1.7 ГГц (28 нм, ARMv7), работающем под управлением ОС Android 4.4.2.

Возможности программы:

- ❖ Создание снимка с помощью стороннего приложения фотокамеры.
- ❖ Сравнение двух изображений.
- ❖ Поиск наиболее схожего изображения из директории к выбранному изображению.
- ❖ Группировка коллекции изображений по степени сходства из выбранной директории.

Настройки программы включают:

- ❖ Выбор одного из методов ORB, BRISK, AKAZE.
- ❖ Сжатие входных изображений - 300, 450 или 600 пикселей на большую сторону.

- Величина порогового значения v_{max}.
- Величина порогового значения $reprojThreshold$.

Вывод результатов сравнения и поиска схожего представлен в виде двух изображений, расположенных рядом и с отображенными, различного цвета, связями, связывающими найденные соответствующие особые точки. В текстовом поле указывается затраченное время, степень сходства, количество связей.

Вывод найденных групп представлен в виде сетки с максимальной шириной в 3 изображения, группы разделены текстовым полем с указанием номера группы. По завершению группировки отображается общее затраченное время.

С интерфейсом программы можно ознакомиться в приложении 2.

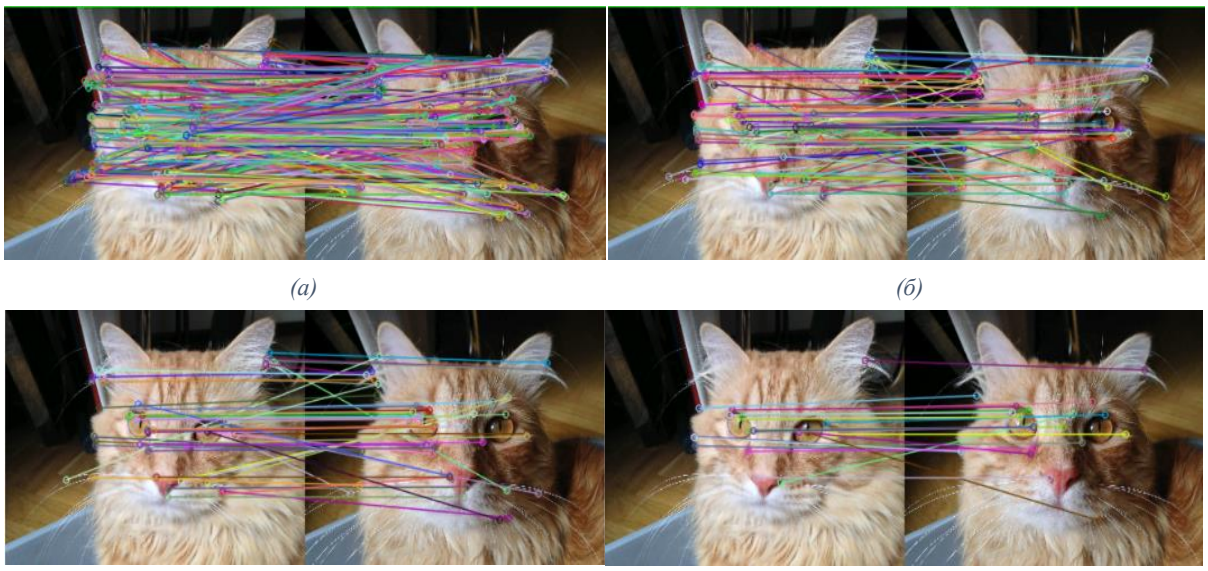
Глава 4: Результаты работы алгоритмов

4.1 Тестирование программы

Приведем пример работы поиска связей и их фильтрации в приложении.

Время замерялось как среднее между тремя запусками. В тестировании применялся метод ORB. Входное изображение масштабируется до 300 пикселей на большую сторону.

На рис. 4.1 изображены найденные методом ближайшего соседа связи, с порогом фильтрации v_{max}. При уменьшении порога остаются наиболее значимые связи.



(а) *(б)*

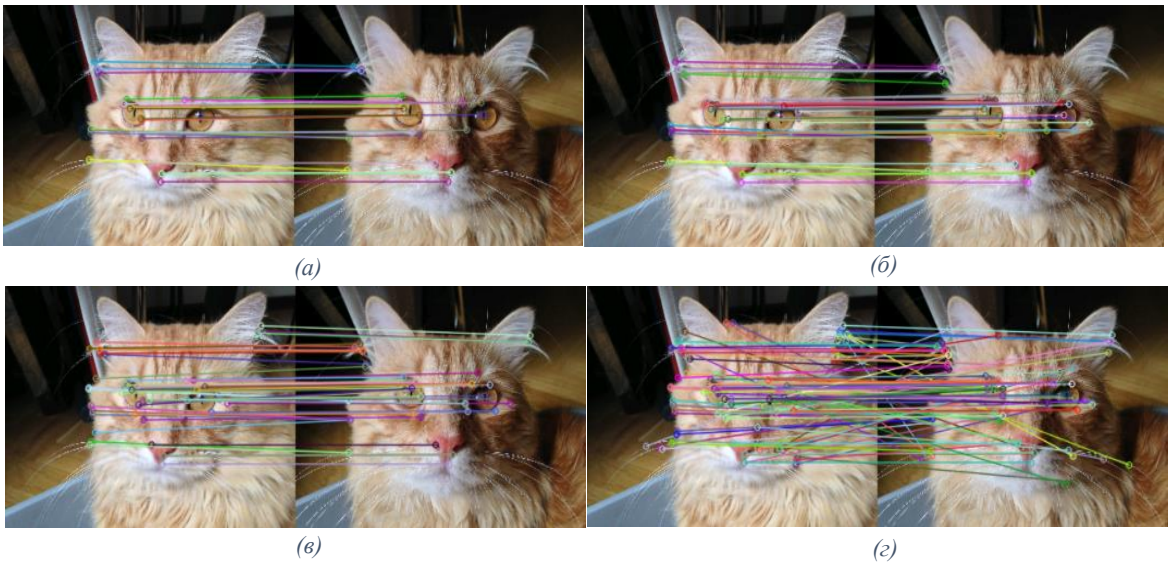
(в) *(г)*

Рис. 4.1 сопоставление дескрипторов по методу ближайшего соседа:

а) Без фильтрации по v_{max}, $|srcPoints'| = 477$. *б)* $v_{max} = 0.9$, $|srcPoints'| = 107$.

в) $v_{max} = 0.85$, $|srcPoints'| = 50$. *г)* $v_{max} = 0.8$, $|srcPoints'| = 22$.

На рис. 4.2 показан результат применения алгоритма RANSAC, с различным порогом $reprojThreshold$. В дальнейшей работе $reprojThreshold = 20$, такое значение обеспечивает необходимый уровень фильтрации связей.

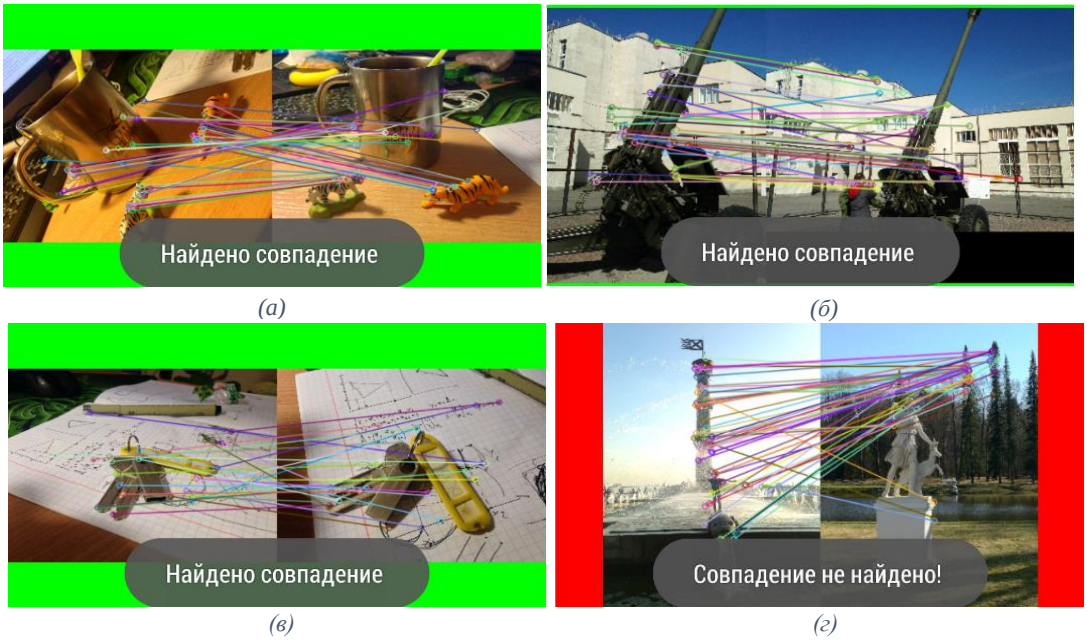


(а) *(б)*

(в) *(г)*

Рис. 4.2 работа фильтра трансформации при $v_{max} = 0.9$:
а) $reprojThreshold = 5$, *связей* $= 26$.
б) $reprojThreshold = 10$, *связей* $= 39$.
в) $reprojThreshold = 20$, *связей* $= 53$.
г) $reprojThreshold = inf$, *связей* $= 107$.

На рис. 4.3 показан результат работы алгоритма сравнения изображений. По результатам тестов данный подход дает верный результат с высокой точностью при вращении, изменении масштаба объекта на изображении и перспективных искажениях.



(а) *(б)* *(в)* *(г)*

Рис. 4.3 Пример работы алгоритма сравнений изображений $\upsilon_{max} = 0.85$:
а) *Связей=63,* $averageDeviation = 0.19$, $alikeValue = 50.6$.
б) *Связей = 44,* $averageDeviation = 0.1$, $alikeValue = 39.6$.
в) *Связей = 29,* $averageDeviation = 0.16$, $alikeValue = 24.2$.
г) *Связей = 59,* $averageDeviation = 0.23$, $alikeValue = 0$.

На рис. 4.4 изображены найденные группы, определенные классификатором после обработки коллекции фотографий. Количество групп, затраченное время и количество ложных совпадений может варьироваться в зависимости от заданного υ_{max} и применяемого метода поиска особых точек и их дескрипторов.

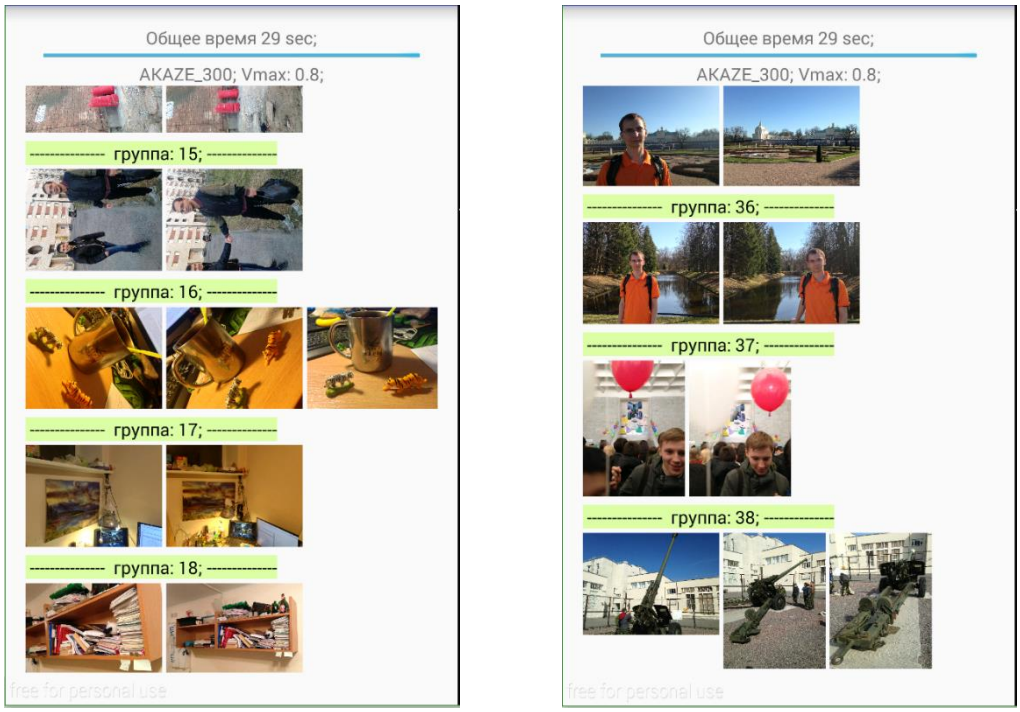


Рис. 4.4 Примеры работы классификатора.

4.2 Сравнительный анализ.

На данном этапе будет проведено сравнение рассматриваемых методов ORB, BRISK, AKAZE основываясь на результатах классификатора. Параметры $\upsilon_{max} = \{0.7, 0.75, 0.8, 0.85, 0.9\}$, $reprojThreshold = 20$, размер входных изображений рассматривается в двух вариантах – сжатие до 300 и 600 пикселей на большую сторону.

Входные данные разделены на следующие группы:

1. 100 фотографий с большим количеством деталей:
    - Высокого качества.
    - С наличием цветного Гауссова шума:
    - Подверженных Гауссову размытию.
2. 30 фотографий текстур.
3. 30 фотографий лиц людей.

Примеры фотографий представлены в приложении 3.

Пояснение к таблицам:

•“KP мс.” – общее затраченное время на поиск особых точек в миллисекундах.

•“кол-во KP”– общее количество найденных особых точек на входном множестве изображений.

•“D мс.” – общее затраченное время на расчет дескрипторов в миллисекундах.

•“Затраты”– $\frac{\text{“}KP\text{ мс.”}+\text{“}D\text{ мс.”}}{\text{“кол–во }KP\text{”}}$ среднее затраченное время на поиск одной особой точки и расчет ее дескриптора.

•“Время”– общее затраченное время на работу программы в секундах, без учета поиска особых точек, расчета дескрипторов и чтения из директории.

•“Групп” – количество определенных групп.

•“Ложные группы”– группы, в которых все фотографии различны.

•“Ошибки в группах”– если есть как минимум 2е схожих фотографии в

группе, то каждая не схожая с этими двумя считается ошибкой.

•“Фотографий без группы” – фотографии, которые не были определены в какую-либо группу.

•“Кол-во связей”– суммарное число связей, по которым были определены группы, связи фотографий без групп не учитываются.

4.1.1 Анализ результатов первой группы

Изображения имеют большое количество деталей и почти все имеют четко выраженный объект на переднем плане, количество групп – 42:

1. 100 фотографий высокого качества.

Результаты тестов приведены в таблице 2.

300					600				
метод	кол-во KP	KP мс.	D мс.	затраты	метод	кол-во KP	KP мс.	D мс.	затраты
ORB	43944	1370	17159	0,4217	ORB	49444	2538	20402	0,464
BRISK	102380	46296	44275	0,8847	BRISK	271132	55831	46917	0,379
AKAZE	23286	5240	5372	0,4557	AKAZE	89031	20623	19775	0,4538

Таблица 1. Влияние размера изображений на работу методов.

Наибольшее количество особых точек найдено с помощью метода BRISK. Метод ORB оказался не сильно чувствительным к изменению размера изображений в выбранных пределах. Наименьшее время на вычисление дескриптора у метода AKAZE.

метод	v_{max}	сжатие	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
ORB	0,8	300	35	39	3	1	12	4002
ORB	0,75	600	35	38	1	0	15	4024
BRISK	0,8	300	115	42	0	0	2	6695
BRISK	0,75	600	550	43	0	0	0	13862
AKAZE	0,8	300	26	38	0	1	14	3315
AKAZE	0,9	600	262	41	0	0	5	14631

Таблица 2. Лучшие результаты работы программы. Фотографии высокого качества.

В данном тесте применение метода BRISK показало лучший результат. Метод ORB, хоть и дает преимущество во времени, но достаточно много

фотографий не определилось в группы. Хорошие результаты с AKAZE, но при меньшем сжатии изображения.

2. 100 фотографий с шумом.

Результаты тестов приведены в таблице 4.

noise 300					noise 600				
метод	кол-во КР	КР мс.	D мс.	затраты	метод	кол-во КР	КР мс.	D мс.	затраты
ORB	43619	1535	15844	0,3984	ORB	49439	4359	18993	0,4723
BRISK	191953	47667	44673	0,4811	BRISK	670454	59870	51488	0,1661
AKAZE	20394	8468	8110	0,8129	AKAZE	78257	22985	20440	0,5549

Таблица 3. Влияние размера изображений на работу методов.

Влияние наличия шума на изображении привело к тому, что метод BRISK определил большое количество особых точек. При этом ORB и AKAZE незначительно изменили этот показатель.

метод	v_{max}	сжатие	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
ORB	0,8	300	34	35	2	2	19	2073
ORB	0,9	600	176	38	0	3	12	4655
BRISK	0,85	300	430	41	0	1	6	4250
BRISK	0,8	600	3264	42	0	1	2	6044
AKAZE	0,8	300	23	34	3	6	20	1996
AKAZE	0,85	600	219	40	0	1	5	6165

Таблица 4. Лучшие результаты работы программы. Фотографии с шумом.

В данном случае лидером является метод AKAZE. Наличие большого количества особых точек у BRISK привело к многократному возрастанию времени, затраченному на фильтрацию ложных связей дескрипторов. ORB снова дает выигрыш во времени, но при меньших результатах.

3. 100 фотографий с размытием:

Результаты тестов приведены в таблице 6.

blur 300					blur 600				
метод	кол-во KP	KP мс.	D мс.	затраты	метод	кол-во KP	KP мс.	D мс.	затраты
ORB	30503	917	12521	0,4405	ORB	21225	1481	9062	0,4967
BRISK	10152	41005	39827	7,9622	BRISK	8874	44933	41727	9,7656
AKAZE	12461	9924	8693	1,494	AKAZE	24655	25382	22549	1,9441

Таблица 5. Влияние размера изображений на работу методов.

При добавлении размытия на изображение у методов BRISK и AKAZE значительно возросли затраты на вычисления, в отличии от ORB. Количество особых точек у BRISK резко уменьшилось.

метод	v_{max}	сжатие	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
ORB	0,75	300	21	34	0	0	26	2246
ORB	0,85	600	32	32	4	4	23	2734
BRISK	0,75	300	6	22	1	0	51	994
BRISK	0,8	600	6	30	4	1	34	1272
AKAZE	0,85	300	17	33	4	6	21	1922
AKAZE	0,8	600	21	38	0	0	14	2936

Таблица 6. Лучшие результаты работы программы. Размытые фотографии.

В работе с размытыми изображениями наилучшей результат при применении AKAZE, худший при BRISK. ORB показывает средние результаты группировки при худшем времени.

4.1.2 Анализ результатов второй группы

30 фотографий с изображением текстур, количество групп – 13.

texture 300					texture 600				
метод	кол-во KP	KP мс.	D мс.	затраты	метод	кол-во KP	KP мс.	D мс.	затраты
ORB	7465	204	2875	0,4125	ORB	8728	732	3635	0,5003
BRISK	33360	13285	12573	0,7751	BRISK	108285	16570	13986	0,2822
AKAZE	3322	1425	1317	0,8254	AKAZE	15675	7320	7215	0,9273

Таблица 7. Влияние размера изображении на работу методов.

Хоть BRISK и обнаружил большое количество особых точек, но

AKAZE, при сопоставимой итоговой точности, оказался значительно быстрее.

метод	v_{max}	сжатие	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
ORB	0,8	300	1	8	1	1	12	792
ORB	0,8	600	1	8	1	0	12	1309
BRISK	0,8	300	9	7	0	0	14	1840
BRISK	0,7	600	72	7	0	0	13	5125
AKAZE	0,7	300	< 1	5	0	0	19	359
AKAZE	0,9	600	6	7	0	0	13	3272

Таблица 8. Лучшие результаты работы программы. Фотографии текстур.

Количество фотографий без групп достаточно велико для всех применяемых алгоритмов. Это связано с особенностями дескрипторов, так как если изображения не имеют четко выраженных уникальных особенностей, то при сравнении дескрипторов, образуются связи между равными по описанию, но не соответствующими по местоположению, особыми точками, в итоге, это приводит к увеличению показателя $averageDeviation$.

4.1.3 Анализ результатов третьей группы

Фотографии лиц людей, количество групп – 11.

face 300					face 600				
метод	кол-во KP	KP мс.	D мс.	затраты	метод	кол-во KP	KP мс.	D мс.	затраты
ORB	12344	286	4642	0,3992	ORB	14490	589	5580	0,4257
BRISK	15064	17326	17959	2,3423	BRISK	34156	15003	12799	0,814
AKAZE	4997	1867	1531	0,68	AKAZE	17770	8105	7052	0,853

Таблица 9. Влияние размера изображении на работу методов.

метод	v_{max}	сжатие	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
ORB	0,9	300	6	12	4	1	5	934
ORB	0,8	600	3	10	2	3	7	317
BRISK	0,9	300	12	9	4	1	10	826
BRISK	0,8	600	13	9	4	1	9	517
AKAZE	0,8	300	1	9	4	0	12	412
AKAZE	0,9	600	18	11	3	2	4	946

Таблица 10. Лучшие результаты работы программы. Фотографии лиц.

Так же, как и с текстурными изображениями, рассматриваемые методы не предназначены для сопоставления лиц людей, в частности для этих целей применяются алгоритмы основывающиеся на пропорциях лица.

С подробными результатами тестов можно ознакомиться в приложении 4.

4.1.4 Контрольное тестирование

По совокупности результатов проведенных тестов лучшая точность работы классификатора достигнута при применении метода AKAZE и сжатии изображений до 600 пикселей на большую сторону.

Далее найдем среднее значение v_{max} для AKAZE и проведем контрольные тестирования на 150 фотографиях, с 60 группами, без наличия “дефектных” фотографий и с набором 20 размытых и 20 с шумом.

метод	v_{max}	сжатие	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	тест
AKAZE	0,9	600	262	41	0	0	5	14631	hight
AKAZE	0,85	600	219	40	0	1	5	6165	noise
AKAZE	0,8	600	21	38	0	0	14	2936	blur
AKAZE	0,9	600	6	7	0	0	13	3272	texture
AKAZE	0,9	600	18	11	3	2	4	946	face
AKAZE	0,87	600	521	58	0	2	8	22720	150
AKAZE	0,87	600	451	57	0	4	12	16016	110x20x20

Таблица 11. Контрольное тестирование метода AKAZE.

Оптимальные параметры программы:

- Метод AKAZE.
- $v_{max} = 0{,}87$.
- Сжатие изображений до 600 пикселей на большую сторону.

Пусть ошибки в группах считаются с весом 2.

✓ Точность на первом контрольном множестве “150”:

$$\left(1 - \frac{2*2+8}{150}\right) * 100\% = 90.66\%;$$

✓ Точность на втором контрольном множестве “110x20x20”:

$$\left(1 - \frac{2*4+12}{150}\right) * 100\% = 86.66\%;$$

Выводы

Метод ORB имеет лучшую скорость в вычислении особых точек и расчета их дескрипторов, что позволяет использовать его в задачах, где необходима обработка изображений в реальном времени. Одной из таких задач является слежение за движущимся объектом. Но высокая скорость работы сказывается на точности сопоставления дескрипторов не в лучшую сторону. Наличие цифрового шума или размытие изображений еще больше ухудшает результат их сравнения.

BRISK отличается от остальных методов тем, что он определяет наибольшее количество особых точек, но, к сожалению, в них попадает и цифровой шум, при этом на фильтрацию образовавшихся ложных связей затрачивается значительное количество времени, хотя итоговая точность достаточно высока. При этом на размытых изображениях было определено малое количество особенностей, что в результате привело к неудовлетворительным показателям работы классификатора, ввиду нехватки данных.

Метод AKAZE проявил себя с лучшей стороны, хоть он и не обладает такой же скоростью как ORB и не имеет такого количества особых точек как BRISK. Но при этом, из-за особенностей его структуры, таких как поиск особых точек на нелинейной многомасштабной пирамиде и описание дескрипторов по трем параметрам, вместо одного, как у ORB и BRISK, получаем высокую точность при сопоставлении изображений и дальнейшего их распределения по группам.

Заключение

В данной работе был проведен сравнительный анализ методов поиска особых точек и их дескрипторов. Для достижения поставленной задачи была разработана программа для мобильных устройств под ОС “Android”.

Данная программа включает в себя:

1. Реализацию методов ORB, BRISK, AKAZE.
2. Сопоставление дескрипторов изображений.
3. Фильтрация полученных связей дескрипторов.
4. Сравнение изображений, по разработанному критерию сходства.
5. Распределение по группам схожих изображений, на основе разработанного алгоритма классификации.

Были проведены сравнительные тесты работы алгоритмов:

1. При различных параметрах алгоритма сопоставления дескрипторов.
2. На двух степенях сжатия изображений.
3. На трех различных группах фотографий.

В результате выделен наилучший метод поиска особых точек и расчета их дескрипторов, определены оптимальное пороговое значение алгоритма сопоставления и необходимый уровень сжатия исходных изображений.

Список литературы.

1. Ethan Rublee, Vincent Rabaud, Kurt Konolige, Gary Bradski: "ORB: an efficient alternative to SIFT or SURF", *Computer Vision (ICCV), IEEE International Conference on. IEEE*, pp. 2564 – 2571, 2011.

2. Stefan Leutenegger, Margarita Chli, Roland Siegwart: "BRISK: Binary Robust Invariant Scalable Keypoints". *Computer Vision (ICCV)*, pp. 2548 – 2555, 2011.

3. Pablo F. Alcantarilla, Jesús Nuevo, Adrien Bartoli: "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces". *In British Machine Vision Conference (BMVC*), 2013.

4. Rosten, Edward, Tom Drummond: "Machine learning for high-speed corner detection", *9th European Conference on Computer Vision (ECCV)*, pp. 430 – 443, 2006.

5. Michael Calonder, Vincent Lepetit, Christoph Strecha, Pascal Fua, "BRIEF: Binary Robust Independent Elementary Features", *11th European Conference on Computer Vision (ECCV)*, pp. 778 – 792, 2010.

6. S. Grewenig, J. Weickert, C. Schroers, A. Bruhn: "Cyclic Schemes for PDE-Based Image Analysis", *In International Journal of Computer Vision, 2013.*

7. X. Yang, K. T. Cheng: "LDB: An ultra-fast feature for scalable augmented reality". *In IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR)*, pp. 49 – 57, 2012.

8. Lowe, David G.: "Object recognition from local scale-invariant features". *Proceedings of the International Conference on Computer Vision*, pp. 1150 – 1157, 1999.

9. Herbert Bay, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features". *Proceedings of the ninth European Conference on Computer Vision*, pp. 404 – 417, 2006.

10. Harris, C., Stephens, M.: "A Combined Corner and Edge Detector". *Proceedings of the 4th Alvey Vision Conference*, pp. 147 – 151, 1988.

11. J. Weickert, H. Scharr.: “A scheme for coherence-enhancing diffusion filtering with optimized rotation invariance”, *Journal of Visual Communication and Image Representation*, pp. 103–118, 2002.

12. Patent SIFT: https://www.google.com/patents/US6711293

13. Patent SURF: http://www.google.com/patents/US20090238460

14. OpenCV: http://opencv.org

15. JSON, A Java serialization/deserialization library: https://github.com/google/gson

Приложение

Приложение 1. Пример работы классификатора

Пусть на вход подано 6 изображений.

- imageN – номер изображения, с которым проводилось сравнение изображения i.
- avD – $averageDeviation$.
- alV – $alikeValue$, степень сходства изображения.

Для заполнения симметричной матрицы классификатора необходимо произвести 15 сравнений, сравнение фотографии с собой не производится (главная диагональ матрицы).

i\j	0	1	2	3	4	5
0	imageN= 0 avD= 100 alV=-1	imageN= 1 avD= 0.22 alV=0	imageN= 2 avD= 0.25 alV=0	imageN= 3 avD= 0.17 alV= 34,27	imageN= 4 avD= 0.81 alV=0	imageN= 5 avD= 0.75 alV=0
1	imageN= 0 avD= 0.22 alV=0	imageN= 1 avD= 100 alV=-1	imageN= 2 avD= 0.34 alV=0	imageN= 3 avD= 0.42 alV=0	imageN= 4 avD= 0.06 alV=30.94	imageN= 5 avD= 0.13 alV= 58,76
2	imageN= 0 avD= 0.25 alV=0	imageN= 1 avD= 0.34 alV=0	imageN= 3 avD= 100 alV=-1	imageN= 3 avD= 0.18 alV=8.76	imageN= 4 avD= 0.53 alV=0	imageN=5 avD=0.68 alV=0
3	imageN= 0 avD= 0.17 alV= 34,27	imageN= 1 avD= 0.42 alV=0	imageN= 2 avD= 0.18 alV=8.76	imageN= 3 avD= 100 alV=-1	imageN= 4 avD= 0.63 alV=0	imageN=5 avD=0.88 alV=0
4	imageN= 0 avD= 0.81 alV=0	imageN= 1 avD= 0.06 alV=30.94	imageN= 2 avD= 0.53 alV=0	imageN= 3 avD= 0.63 alV=0	imageN= 4 avD= 100 alV=-1	imageN=5 avD=0.19 alV=7.15
5	imageN= 0 avD= 0.75 alV=0	imageN= 1 avD= 0.13 alV= 58,76	imageN=2 avD=0.68 alV=0	imageN=3 avD=0.88 alV=0	imageN=4 avD=0.19 alV=7.15	imageN=5 avD=100 alV=-1

Таблица 1. Матрица результатов попарных сравнений изображений.

Далее каждая строка сортируется по убыванию $alikeValue$:

0	imageN= 3 avD= 0.17 alV= 34,27	imageN= 1 avD= 0.22 alV=0	imageN= 2 avD= 0.25 alV=0	imageN= 4 avD= 0.81 alV=0	imageN= 5 avD= 0.75 alV=0	imageN= 0 avD= 100 alV=-1
1	imageN= 5 avD= 0.13 alV= 58,76	imageN= 4 avD= 0.06 alV=30.94	imageN= 0 avD= 0.22 alV=0	imageN= 2 avD= 0.34 alV=0	imageN= 3 avD= 0.42 alV=0	imageN= 1 avD= 100 alV=-1
2	imageN= 3 avD= 0.18 alV=8.76	imageN= 0 avD= 0.25 alV=0	imageN= 1 avD= 0.34 alV=0	imageN= 4 avD= 0.53 alV=0	imageN=5 avD=0.68 alV=0	imageN= 2 avD= 100 alV=-1
3	imageN= 0 avD= 0.17 alV= 34,27	imageN= 2 avD= 0.18 alV=8.76	imageN= 1 avD= 0.42 alV=0	imageN= 4 avD= 0.63 alV=0	imageN=5 avD=0.88 alV=0	imageN= 3 avD= 100 alV=-1
4	imageN= 1 avD= 0.06 alV=30.94	imageN=5 avD=0.19 alV=7.15	imageN= 0 avD= 0.81 alV=0	imageN= 2 avD= 0.53 alV=0	imageN= 3 avD= 0.63 alV=0	imageN= 4 avD= 100 alV=-1
5	imageN= 1 avD= 0.13 alV= 58,76	imageN=4 avD=0.19 alV=7.15	imageN= 0 avD= 0.75 alV=0	imageN=2 avD=0.68 alV=0	imageN=3 avD=0.88 alV=0	imageN=5 avD=100 alV=-1

Таблица 2. Сортировка по значению $alikeValue$.

Сгруппируем изображения:

0. Для изображения с номером $i = 0$ в первой ячейке $i_1 = 3$, у $i = 3 - i_1 = 0 =>$ номера 0 и 3 формируют группу 1.

1. Для $i = 1 - i_1 = 5$, а для $i = 5 - i_1 = 1 =>$ номера 1 и 5 формируют группу 2.

2. Для $i = 2 - i_1 = 3$, но у $i = 3 - i_1 = 0$, рассмотрим значение $averageDeviation$ между 2 и 0: avD $> 0.2 =>$ для изображения 2 группы нет.

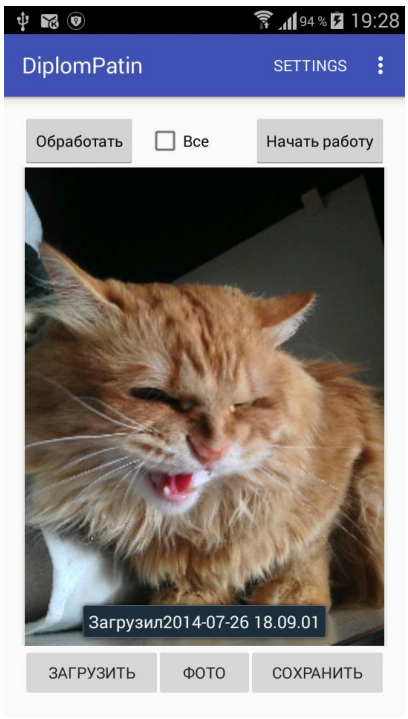
3. $i = 3$ не рассматривается, т.к. группа уже есть.

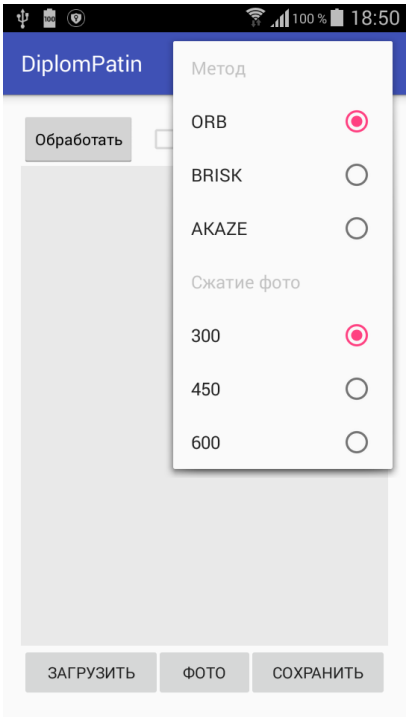
4. Для $i = 4 - i_1 = 1$, но у $i = 1 - i_1 = 5$, рассмотрим значение $averageDeviation$ между 4 и 5: avD $< 0.2 =>$ изображение 4 добавляется в группу 2.

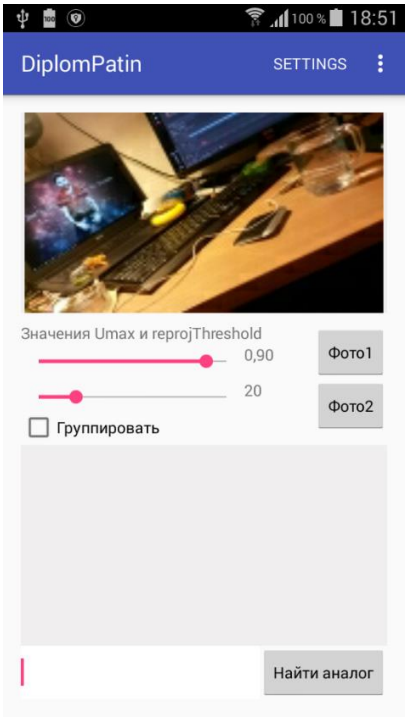
5. $i = 5$ не рассматривается, т.к. группа уже есть.

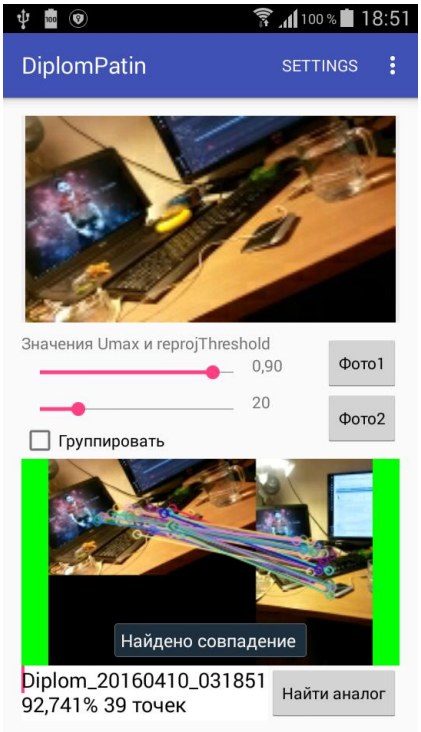
Приложение 2. Интерфейс программы

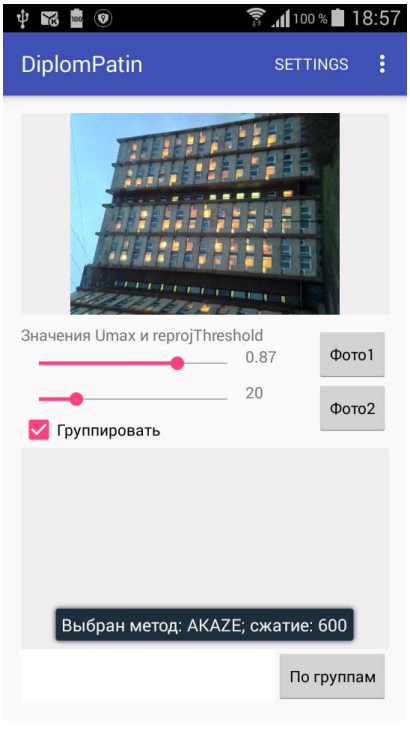
Интерфейс программы на смартфоне Samsung Galaxy S4 mini.





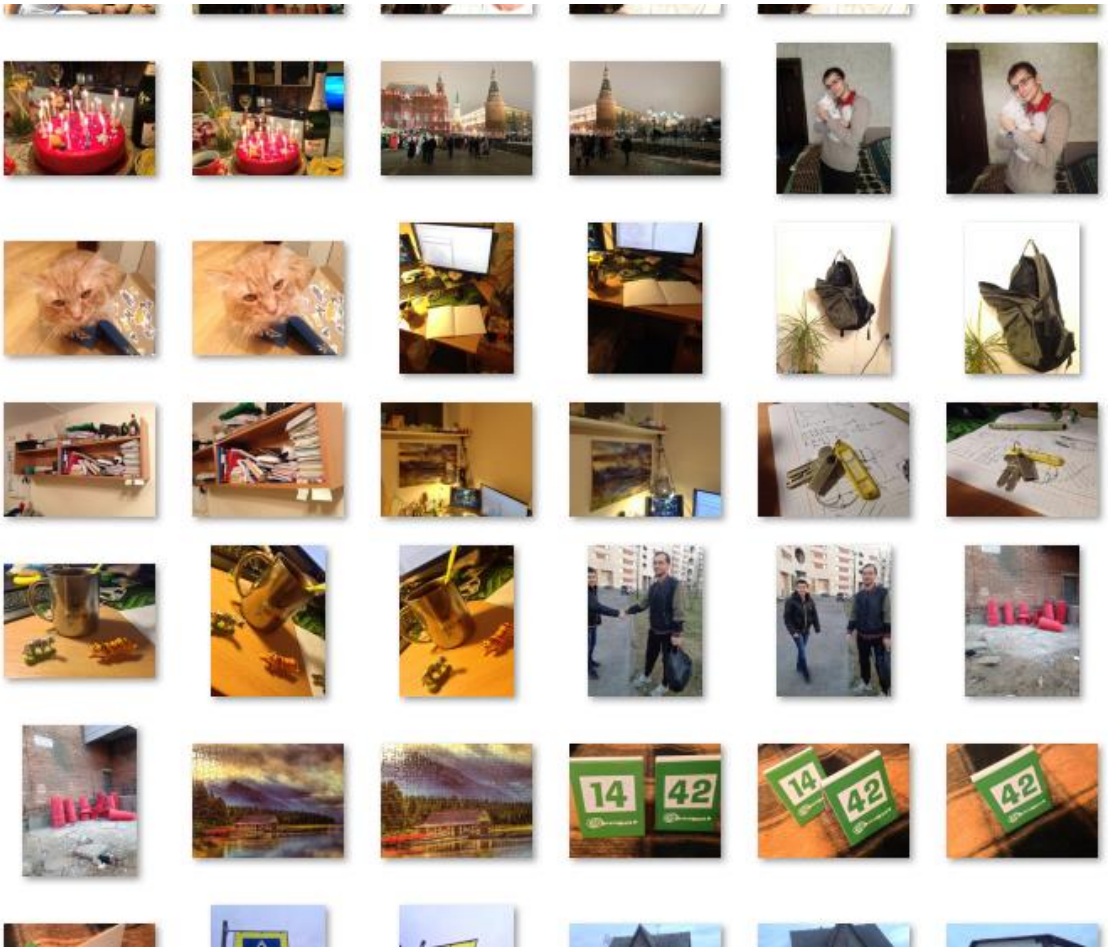






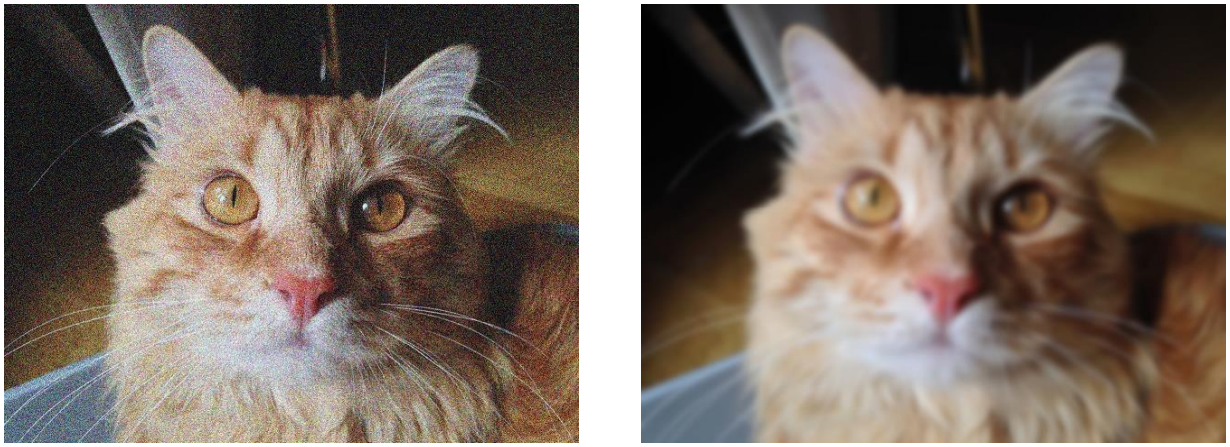
Приложение 2. Примеры изображений

Группа 1:



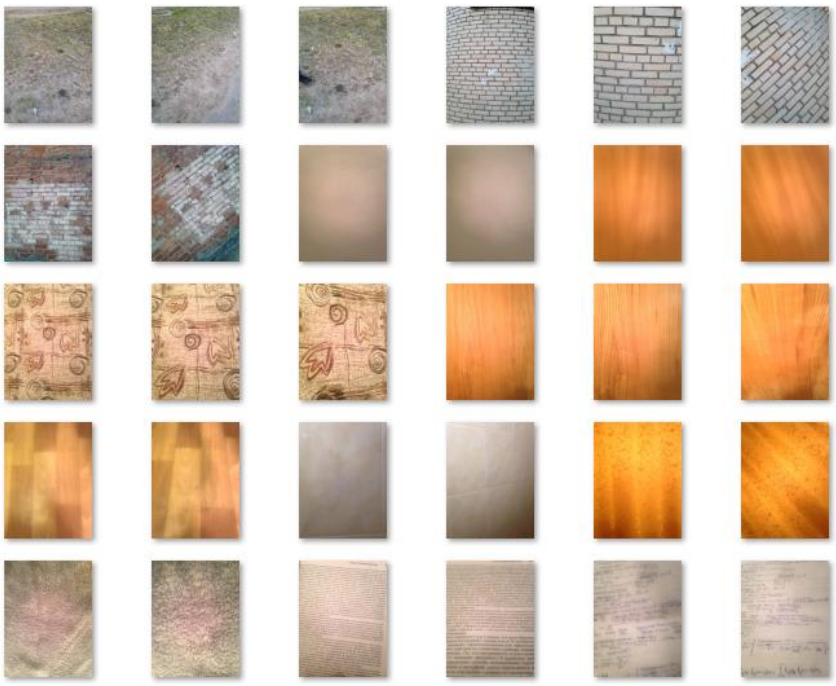
Фотографии с большим количеством ярко выраженных деталей.

Цифровой шум и размытие:



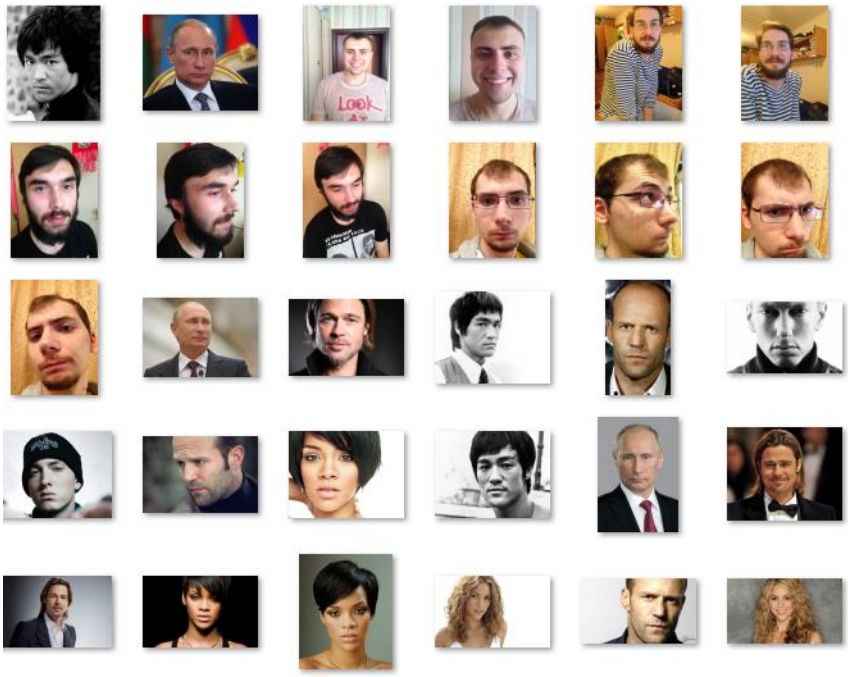
Справа пример добавления цифрового шума. Слева пример размытия фотографии.

Группа 2:



Фотографии текстурного содержания.

Группа 3:



Фотографии лиц.

Приложение 3. Подробные результаты тестов

1. Первая группа.

Тесты изображений высокого качества.

hieght ORB 300						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	30	30	0	0	34	2256
0,75	31	34	1	0	24	2945
0,8	35	39	3	1	12	4002
0,85	45	35	2	12	9	5147
0,9	85	33	2	4	23	6846

hieght ORB 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	34	33	1	0	27	3264
0,75	35	38	1	0	15	4024
0,8	40	38	1	2	13	5011
0,85	66	38	1	3	10	6159
0,9	158	38	1	2	11	7656

hieght BRISK 300						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	102	38	0	0	14	3940
0,75	103	41	0	0	5	5150
0,8	115	42	0	0	2	6695
0,85	174	43	0	1	2	8501
0,9	257	39	0	1	8	11082

hieght BRISK 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	524	40	0	0	9	11143
0,75	550	43	0	0	0	13862
0,8	599	43	1	0	1	16989
0,85	670	41	0	0	3	21028
0,9	696	42	0	1	3	25480

hieght AKAZE 300						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	17	36	4	0	19	2301
0,75	20	36	2	2	16	2704
0,8	26	38	0	1	14	3315
0,85	48	36	2	4	17	3898
0,9	94	38	2	2	13	4798

hieght AKAZE 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	98	38	1	2	8	7549
0,75	120	39	0	0	11	8757
0,8	171	41	1	1	4	10300
0,85	231	39	0	2	5	12261
0,9	262	41	0	0	5	14631

Тесты изображений с цифровым шумом.

noise ORB 300						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	28	25	0	0	45	1424
0,75	32	32	2	0	28	1566
0,8	34	35	2	2	19	2073
0,85	46	38	3	4	12	2845
0,9	92	36	4	4	17	4308

noise ORB 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	36	27	0	0	40	2022
0,75	37	35	2	0	22	2095
0,8	42	41	4	0	9	2439
0,85	71	38	1	5	10	3181
0,9	176	38	0	3	12	4655

noise BRISK 300						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	295	30	0	0	33	1947
0,75	298	37	0	0	18	1949
0,8	324	39	0	2	9	2808
0,85	430	41	0	1	6	4250
0,9	475	40	2	0	11	6718

noise BRISK 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	3180	38	0	0	14	4136
0,75	3173	41	0	0	7	4446
0,8	3264	42	0	1	2	6044
0,85	3165	42	0	0	6	8785
0,9	3225	40	2	0	10	12792

noise AKAZE 300						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	15	30	4	2	34	1380
0,75	17	33	5	0	29	1641
0,8	23	34	3	6	20	1996
0,85	41	33	4	5	25	2470
0,9	84	36	6	1	22	3161

noise AKAZE 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	80	37	1	1	15	3536
0,75	103	39	1	1	8	4236
0,8	160	38	0	2	10	5046
0,85	219	40	0	1	5	6165
0,9	251	39	1	1	7	7342

Тесты размытых изображений.

blur ORB 300							blur ORB 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	20	27	0	0	41	1900	0,7	14	28	3	0	38	1455
0,75	21	34	0	0	26	2246	0,75	15	35	8	1	22	1796
0,8	24	37	2	4	16	2558	0,8	18	36	8	1	17	2124
0,85	35	37	4	4	16	3487	0,85	32	32	4	4	23	2734
0,9	72	34	4	5	20	4689	0,9	80	35	7	3	19	3326

blur BRISK 300							blur BRISK 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	6	19	0	0	57	753	0,7	5	18	0	0	60	788
0,75	6	22	1	0	51	994	0,75	5	22	1	0	51	947
0,8	6	31	8	1	33	1305	0,8	6	30	4	1	34	1272
0,85	9	30	10	6	28	1557	0,85	9	31	4	4	29	1482
0,9	17	29	4	9	25	1956	0,9	25	31	6	7	25	1696

blur AKAZE 300							blur AKAZE 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	8	30	6	2	34	1387	0,7	17	31	0	0	31	2329
0,75	9	31	8	1	30	1477	0,75	18	37	0	0	22	2643
0,8	11	34	7	5	21	1683	0,8	21	38	0	0	14	2936
0,85	17	33	4	6	21	1922	0,85	52	36	1	8	9	3438
0,9	35	33	7	6	21	2557	0,9	127	38	1	2	12	4103

2. Вторая группа.

Тесты фотографий текстур.

texture ORB 300							texture ORB 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	1	7	2	0	15	431	0,7	1	8	1	0	12	1089
0,75	1	7	3	0	15	555	0,75	1	8	1	0	12	1089
0,8	1	8	1	1	12	792	0,8	1	8	1	0	12	1309
0,85	2	8	3	1	12	1030	0,85	3	9	3	1	9	1632
0,9	4	8	1	0	13	1390	0,9	8	8	0	2	11	2285

texture BRISK 300							texture BRISK 300						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	9	5	0	0	19	994	0,7	9	5	0	0	19	994
0,75	9	6	0	0	17	1371	0,75	9	6	0	0	17	1371
0,8	9	7	0	0	14	1840	0,8	9	7	0	0	14	1840
0,85	11	7	2	0	15	2188	0,85	11	7	2	0	15	2188
0,9	13	6	2	0	13	3875	0,9	13	6	2	0	13	3875

texture AKAZE 300							texture AKAZE 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	< 1	5	0	0	19	359	0,7	2	6	0	0	15	1947
0,75	< 1	4	0	0	21	389	0,75	3	6	0	0	15	2236
0,8	< 1	4	0	0	21	447	0,8	4	6	0	0	15	2577
0,85	1	5	1	0	19	547	0,85	5	6	0	0	15	2926
0,9	1	7	2	0	15	657	0,9	6	7	0	0	13	3272

3. Третья группа.

Тесты фотографий лиц.

face ORB 300							face ORB 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	2	2	1	0	26	43	0,7	2	3	0	0	24	90
0,75	2	7	4	0	16	212	0,75	3	7	3	0	16	209
0,8	3	8	4	2	10	279	0,8	3	10	2	3	7	317
0,85	3	11	4	2	5	503	0,85	6	9	4	3	7	466
0,9	6	12	4	1	5	934	0,9	13	8	3	0	14	795

face BRISK 300							face BRISK 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	3	1	0	0	28	93	0,7	10	5	1	0	19	254
0,75	3	6	3	0	18	228	0,75	11	7	3	0	14	385
0,8	4	11	5	0	8	350	0,8	13	9	4	1	9	517
0,85	5	7	2	7	6	499	0,85	20	8	2	3	11	726
0,9	12	9	4	1	10	826	0,9	24	9	4	3	8	1094

face AKAZE 300							face AKAZE 600						
v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей	v_{max}	время сек.	групп	ложных групп	ошибок в группах	фотографий без группы	кол-во связей
0,7	< 1	5	3	1	19	222	0,7	4	10	5	0	10	341
0,75	1	8	7	1	12	327	0,75	4	11	4	1	7	458
0,8	1	9	4	0	12	412	0,8	6	10	4	2	7	532
0,85	2	8	2	3	10	507	0,85	12	11	3	4	2	670
0,9	4	10	4	2	7	687	0,9	18	11	3	2	4	946